Numerical Calculation of Molecular Surface Area. II. Speed of Calculation

ANDREY A. BLIZNYUK* and JILL E. GREADY†

Department of Biochemistry, University of Sydney, Sydney, NSW 2006, Australia

Received 23 January 1995; accepted 3 August 1995

ABSTRACT

A new fast and accurate algorithm for numerical calculation of the van der Waals and solvent accessible surface areas based on the sorted table of cosines is described. This algorithm does not depend upon the particular distribution of the points on the sphere surface, and thus allows use of the most accurate points distribution available. Direct comparisons (on the same computer) with other known fast algorithms are performed. The comparisons show that this algorithm is the fastest when accurate calculation of the van der Waals surface is required and is at least as fast as the fastest competitor algorithm for the evaluation of the solvent accessible surface area. © 1996 by John Wiley & Sons, Inc.

Introduction

As computer speeds increase, more realistic models are being used to investigate the behavior of molecular systems. Inclusion of a solvent into calculations is, usually, required for better description of chemical or biochemical processes. However, as the use of explicit solvent molecules greatly increases the computation time, simplified models using surface area calculations for solvation energy have been developed (see refs. 1–6 and references therein). Hence, new methods for calculating molecular surface are currently attracting much attention from researchers in different fields of chemistry and biochemistry.

With development of tools similar to SCULPT,[7] which allow investigation of dynamics and interactions of biomolecules in real time, the fast calculation of molecular surface or its approximation becomes a matter of prime importance. Because analytical approaches are not, in general, suitable for this purpose, numerical methods have received widespread attention. Recently, three articles proposing modifications of the Shrake and Rupley approach[8] have been published.[2–4] The first article[2] describes a very fast, but approximate, method for surface area estimation, which is likely to be suitable only for calculations where accuracy is not critical. The second paper[3] describes a method suitable only for specific points distributions, which are far from ideal.[4] As we have shown

*On leave from Novosibirsk Institute of Bioorganic Chemistry, Novosibirsk 630090, Russia.

†Author to whom all correspondence should be addressed at Division of Biochemistry and Molecular Biology, John Curtin School of Medical Research, Australian National University, Canberra, ACT 0200, Australia.

CCC 0192-8651 / 96 / 080970-06

in our previous article,[1] the choice of points distribution is critical for the accuracy of the surface area evaluation. The algorithm of ref. 3 has since been modified to work with any points distribution.[4] Finally, a very fast, simple, and elegant method for evaluation of molecular surface area has been defined in ref. 4 and implemented in the NSC program.[4] Comparison of the method modified from ref. 3 with the new algorithm in ref. 4 showed that the latter is approximately two times faster. This method[4] is claimed to be at least as fast as the algorithm of Le Grand and Merz,[2] but no direct comparison has been made. Also, from the description of the algorithm, it is not clear how fast it will be in the calculation of the van der Waals (VDW) surface,[9] which is widely used in boundary element methods for estimation of the solvation energy.[5,6,10]

In this article we describe an alternative approach to calculation of the molecular surface area. This method is approximately as fast as the algorithm[4] for calculation of the solvent accessible (SA) surface area[9] and much faster for the VDW surface calculations. Also, direct comparison of the methods of Le Grand and Merz,[2] the algorithm of ref. 4, and our approach is reported.

Method

Algorithms for VDW or SA surface area calculations assume that the molecule can be represented as a collection of fused spheres. Each atom (or group of atoms) is modeled as a sphere of fixed radius. In numerical algorithms it is also assumed that each sphere can be represented by a number of points on its surface. Following the commonly used approach,[1-4,8] we assume that the number of points on the surface of each sphere and their distribution are the same. Thus, to specify completely the surface of a molecule, we need to know the coordinates of the atoms, their radii (R_i), the number of points (N) on the sphere surface, and their coordinates on the surface of a unit sphere. The numerical algorithm systematically removes points that lie within the volume of another sphere. In Figure 1, the points **z** with distance |**p**–**z**| less than distance |**p**–**f**| will be discarded, or equivalently, the points **z** for which cos(**p**–**i**–**z**) > cos(**p**–**i**–**f**) will be discarded.

The original approach[8] analyzed which points are accessible for each intersection of atoms **i** and **j**. This method may be divided into two separate parts: (i) finding which atoms intersect and (ii) finding which points should be marked as inaccessible. The first part is similar to the problem of calculation of the electrostatic energy with cutoff, as present in molecular mechanics calculations. The usual approach to reduce computing time is to divide the space occupied by the molecule into cubes with the dimension of the cube equal to the cutoff distance (see, for example, ref. 11). Only atoms in the same or adjacent cubes can intersect. The modification of this approach to the problem of molecular surface calculations is described in ref. 4. This solution is general and can be applied to any method of points removal. Usually, only part (ii) is considered as a surface area algorithm (see, for example, ref. 2).

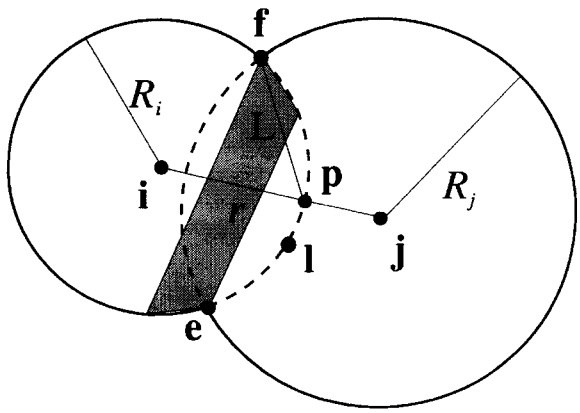

FIGURE 1. Schematic representation of the intersection of the two atom-centered spheres. R_i and R_j are the radii of the spheres; r is the interatomic distance. Point **p** is the result of intersection of line **i**–**j** and sphere **i**. L is the distance from **p** to the circle of intersection; **l** is the nearest surface point to **p**. Points **e** and **f** belong to the intersection circle and lie on the plane (**p**, **l**, **i**). Only points in the shadowed area need to be checked explicitly by calculating the scalar product (**j**–**i**) · **z** (see text).

To check if the point **z** (any point) on the surface of atom **i** should be removed due to intersection with atom **j**, it is necessary to calculate a scalar product of vectors (**j**–**i**) and **z** which is equal to $r * \cos(\mathbf{p}–\mathbf{i}–\mathbf{z})$, where r is distance **i**–**j** (see Fig. 1). If this product is greater than $r * \cos(\mathbf{p}–\mathbf{i}–\mathbf{f})$, then this point should be marked as inaccessible. The last term can easily be calculated using eq. (1):

$$r * \cos(\mathbf{p}–\mathbf{i}–\mathbf{f}) = r * \cos(\mathbf{j}–\mathbf{i}–\mathbf{f}) = \left(r^2 + R_i^2 - R_j^2\right)/(2R_i) \quad (1)$$

To our knowledge, this approach was first described in ref. 4 and is slightly faster to compute than the frequently used square-distance calculation. Obviously, the fewer scalar products we need to evaluate, the faster the algorithm.

The authors of ref. 4 suggested a change in the order of the loops, i.e., the loop over the points **z** of atom **i** precedes the loop over the neighboring atoms **j**. The rationale is that most of the surface points are buried by a few nearest atoms. For calculation of SA surface area, where a surface point has high probability of being buried, this algorithm should be fast. However, for evaluation of the VDW surface area, where about half the points on the surface are accessible, such a method should not be significantly faster compared with the standard algorithm.

We will follow the usual approach, namely, the loop over the neighboring atoms precedes the loop over the surface points, and will demonstrate how the number of scalar products $(\mathbf{j}-\mathbf{i}) \cdot \mathbf{z}$ may be reduced to a minimum.

Suppose there is an intersection of spheres **i** and **j** with radii R_i and R_j. Let us define point **p** at the intersection of the surface of sphere **i** and the line connecting the centers of **i** and **j** (see Fig. 1). Point **l** is the nearest surface point to **p**. Line **e**–**f** is the result of intersection of the plane (**p**, **l**, **i**) and the plane of spheres intersection. Let us assume that distance $|\mathbf{l}-\mathbf{e}|$ is less than distance $|\mathbf{l}-\mathbf{f}|$. Clearly all points **z**, with the distance $|\mathbf{l}-\mathbf{z}| > |\mathbf{l}-\mathbf{f}|$ should be retained and all points **z**, with the distance $|\mathbf{l}-\mathbf{z}| < |\mathbf{l}-\mathbf{f}|$ should be removed. So, we need to check only points **z**, for which the condition $|\mathbf{l}-\mathbf{e}| < |\mathbf{l}-\mathbf{z}| < |\mathbf{l}-\mathbf{f}|$ holds. Instead of using distances, we will use the cosines, which are slightly faster to calculate. The above condition then becomes $\cos(\mathbf{l}-\mathbf{i}-\mathbf{e}) > \cos(\mathbf{l}-\mathbf{i}-\mathbf{z}) > \cos(\mathbf{l}-\mathbf{i}-\mathbf{f})$. The corresponding cosines can easily be calculated. Indeed, the equation for the plane (**p**, **l**, **i**) (**i** = 0) is

$$\mathbf{v} \cdot \mathbf{X} = 0 \qquad \text{where } \mathbf{v} = (\mathbf{j}-\mathbf{i})/r \times \mathbf{l} \quad (2)$$

The equation for the plane of spheres intersection is

$$(\mathbf{j}-\mathbf{i})/r \cdot \mathbf{X} = \left(r^2 + R_i^2 - R_j^2\right)/(2rR_i) = d \quad (3)$$

The line of intersection of these planes is

$$\mathbf{b} + \mu\mathbf{c}$$
$$\text{where } \mathbf{c} = (\mathbf{j}-\mathbf{i})/r \times \mathbf{v} \text{ and } \mathbf{b} = (\mathbf{j}-\mathbf{i})/r * d \quad (4)$$

The value μ can be found using eq. (5), because points **e** and **f** are on the surface of a unit sphere:

$$(\mathbf{b} + \mu\mathbf{c})^2 = 1 \quad (5)$$

Using simple vector and scalar products transformations, μ becomes

$$\mu = \pm\text{sqrt}[(1 - a^2)(1 - d^2)]$$
$$\text{where } a = (\mathbf{j}-\mathbf{i})/r \cdot \mathbf{l} \quad (6)$$

Finally, the necessary cosines are

$$\cos(\mathbf{l}-\mathbf{i}-\mathbf{f}) = (\mathbf{j}-\mathbf{i}) \cdot (\mathbf{i}-\mathbf{f}) = a * d - \mu$$
$$\cos(\mathbf{l}-\mathbf{i}-\mathbf{e}) = (\mathbf{j}-\mathbf{i}) \cdot (\mathbf{i}-\mathbf{e}) = a * d + \mu \quad (7)$$

It should be noted, that this approach requires calculation of only the square of the interatomic distance (r^2). Therefore, only one square root per intersection is needed in eq. (7). It is important to choose the nearest point to **p** as point **l**, because in that case the size of the sector containing points which satisfy the condition $\cos(\mathbf{l}-\mathbf{i}-\mathbf{e}) > \cos(\mathbf{l}-\mathbf{i}-\mathbf{z}) > \cos(\mathbf{l}-\mathbf{i}-\mathbf{f})$ is minimal (shadowed sector in Fig. 1). Hence, only the minimum number of scalar products $(\mathbf{j}-\mathbf{i}) \cdot \mathbf{z}$ need to be evaluated.

Taking into account that the coordinates of points on the unit sphere do not change during the calculation, it is useful to precalculate $\cos(\mathbf{l}-\mathbf{i}-\mathbf{z})$ for every possible pair **l**–**z**. We constructed a two-dimensional table. Each element of this table contains the point number and the cosine. For example, the first row of this table contains cosines between the first point and all the other points; the second row holds cosines between the second point and all other points, and so on. The elements of each row are sorted in descending order of cosines. This table form greatly simplified the search for points to be removed. Suppose the nearest surface point to **p** is the point with number 10. In this case we select the numbers and cosines of points from the tenth row. We will mark all points as inaccessible until we find in this row a cosine smaller than $\cos(\mathbf{l}-\mathbf{i}-\mathbf{e})$.

In order to find the number of the point nearest to **p**, we used a look-up table similar to the look-up table described in ref. 2. We determined the surface point nearest to a large number of spherical polar coordinate positions on the sphere and stored this information. For each intersection we calculated the spherical polar coordinates of the point **p** and selected the number of the nearest point from the look-up table.

We can expect that some of the surface points of atom **i** which need to be checked explicitly [calcu-

lation of the dot product $(\mathbf{j}-\mathbf{i}) \cdot \mathbf{z}]$ for the intersection of atoms **i** and **j** may not require such examination for the intersection of atoms **i** and **k**. In our algorithm we perform the explicit test only after removing all points which do not require such a test. For example, if atom **i** has intersections with atoms **a**, **b**, and **c**, we examine the intersections **i**–**a**, **i**–**b**, and **i**–**c** first and remove all points which do not require the explicit test. Then we make a loop over these intersections again and test points which are not yet marked as inaccessible and require explicit examination. This approach proved to be faster than doing all the tests in one loop.

Because the evaluation of the scalar products $(\mathbf{j}-\mathbf{i}) \cdot \mathbf{z}$ is very time consuming, the authors of ref. 2 elected to ignore it completely. They suggested use of the nearest point **l** instead of the true point of intersection **p**. Such a procedure introduces additional inaccuracy to the numerical method for the evaluation of the molecular surface. Figure 2 compares the average deviation of the VDW surface area calculation by the algorithm of ref. 2 and by our algorithm, described above, for the same number of points. The data in Figure 2 have been obtained by the technique described in our previous article.[1] As can be seen from Figure 2, the average deviations in the calculation of molecular surface are up to three times larger for the algorithm of ref. 2. This renders this approach suitable only for rough approximation of the molecular surface area. It should also be noted that increasing the number of points in this algorithm does not necessarily lead to an increase in accuracy.

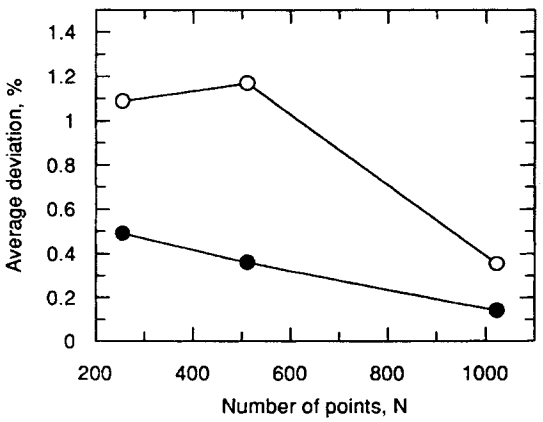

FIGURE 2. Comparison of the accuracy of the VDW surface area calculation for our algorithm (●) and the algorithm of ref. 2 (○). The accuracy of the VDW surface area evaluation has been estimated as an average relative deviation R: $R = (100\% / N_0)\Sigma_i^{N_0} R_i$, where $R_i = |A_i(\text{calculated}) - A_i(\text{exact})| / A_i(\text{exact})$. N_0 in the above formula is the number of molecules in the test set (197) and A_i is the VDW surface area of molecule i (see ref. 1 for more details).

On the other hand, the algorithm of ref. 2 does not require evaluation of either scalar products or the value of μ [eq. (6)], which makes it fast. To achieve an even higher speed in calculations, the authors of ref. 2 suggested use of binary strings to represent the status of each point on the surface of the sphere (buried/accessible). The usefulness of binary strings is that it allows use of the very fast logical AND operator between the binary string for sphere **i** and the binary mask to mark points as buried. The mask should contain 0 for points changing their status to inaccessible and 1 otherwise. The form of the table of masks suggested in ref. 2 is the two-dimensional array of binary strings. The first dimension is the point number $(1-N)$; the second is the distance $|\mathbf{p}-\mathbf{e}|$ (0–2). This table should have finite dimensions, so we have to use discrete values for distances $|\mathbf{p}-\mathbf{e}|$, using arbitrary steps. The value of 0.01 has been utilized in ref. 2. The size of the mask table in this case is $N * 200 * N/8$ bytes ($N/8$ is the size of the binary string).

In our implementation of this algorithm we used a slightly different form of the table of masks. The second dimension is the value of d [eq. (3)]. Only masks corresponding to positive values of d (0–1) are stored. If during calculation, d becomes negative, then the mask of the other side of the sphere will be selected and its binary complement will be used. The value of d may be negative only if the interatomic distance becomes smaller than $\text{sqrt}(R_i^2 - R_j^2)$. Such a situation, if we choose R_i and R_j to be the van der Waals radii, may occur for S—H bonds, which are rare in proteins. This choice of the form of the mask's table reduces the memory requirement by half, without loss in speed.

The algorithms described above (ours and that of Le Grand and Merz[2]) have been implemented in C language. The programs are available from the authors on request.

Results and Discussion

It is difficult to compare the speed of the different algorithms. The computation time will depend on the reference machine (instruction set, cache size, quality of translation), the input data, and, of course, on the implementation. We decided to perform all calculations on the personal IRIS 4D/35,

because computers of this series have previously been used for such tests.[2,4]

The choice of test molecules for the speed evaluations has been guided by two factors. First, the geometry of compounds should be readily available. Second, the molecules should be large enough to allow accurate measurements of execution time. Because our main field of interest is the evaluation of the molecular surface of biopolymers, three molecules have been selected from the Brookhaven Protein Data Bank.[12] The trypsin inhibitor (4pti) contains 454 nonhydrogen atoms, dihydrofolate reductase (1drf) contains 1497 nonhydrogen atoms, and ribulose 1,5-bisphosphate carboxylase/oxygenase (8rub) contains 4666 nonhydrogen atoms. Two types of molecular surfaces have been calculated for these enzymes: van der Waals molecular surface, with radii taken from ref. 13, and solvent accessible surface with van der Waals radii increased by 1.4 Å for the probe radius. The number of sphere–sphere intersections are 1490, 4823, and 15011 for the VDW surface computation and 8292, 28547, and 94375 for the SA surface area calculation, respectively.

The average times of 10 runs on the IRIS 4D/35 are given in Table I. Only the time needed for execution of the algorithm itself is shown, because the generation of points (NSC program) or precalculation of the look-up tables (our algorithm and the algorithm of ref. 2) do not depend on the input data and can be done before the calculations. Formula (8) has been used for the calculation of the molecular surface areas because only this formula is implemented in the program NSC[4]:

$$S = \sum_i 4\pi R_i^2 M_i / N \qquad (8)$$

where M_i is the number of accessible points on sphere i and the summation is over all atoms (spheres) in the molecule.

The cubic lattice method has been used throughout all calculations in the form described in ref. 4 to avoid unnecessary checking of sphere–sphere interactions (see the beginning of the Methods section).

The fastest algorithm is that of Le Grand and Merz.[2] Our algorithm and the approach of ref. 4 give similar results for SA surface area calculations, although our method tends to be faster as the number of points increases. Our method is clearly superior to the algorithm of ref. 4 when calculating the VDW surface area. About 70% of atoms are inaccessible when SA surface area is calculated. Hence, the probability of a surface point becoming buried is high. This greatly reduces the number of sphere–sphere intersections tested in the inner loop of the algorithm of ref. 4 and increases its speed. On the other hand, nearly all atoms have nonzero surface area in the VDW surface calculations, even though many are buried in the core of a compact protein molecule. This requires almost all sphere–sphere interactions to be executed in the inner loop of the algorithm of ref. 4 and, therefore, greatly reduces its speed. The speed

TABLE I.
Time (in seconds) of Surface Area Calculation for Three Proteins on an SGI IRIS 4D / 35 Workstation for van der Waals (VDW) and Solvent Accessible (SA) Surface Areas.

		VDW			SA		
Points	Algorithm[a]	4pti	1drf	8rub	4pti	1drf	8rub
256	Ref. 2	0.10	0.23	1.12	0.40	1.46	4.95
252	Ref. 4	0.52	1.69	5.29	0.81	2.66	8.65
256	This work	0.21	0.71	2.24	0.78	2.75	9.04
512	Ref. 2	0.12	0.40	1.28	0.48	1.71	5.82
492	Ref. 4	0.94	3.07	9.56	1.24	3.94	12.28
512	This work	0.31	1.03	3.24	1.10	3.85	12.64
1024	Ref. 2	0.17	0.55	1.72	0.70	2.48	8.39
1082	Ref. 4	2.01	6.36	19.92	2.19	6.80	21.02
1024	This work	0.50	1.66	5.20	1.72	6.02	19.69
1962	Ref. 4	3.48	11.40	35.14	3.55	11.03	32.98
2048	This work	0.93	2.87	8.97	2.98	10.14	32.96

[a]Due to large memory requirements (50 MB) of the algorithm of ref. 2 with 2048 points, only the algorithm of ref. 4 and ours are compared for this number of points.

of our algorithm is determined by the number of sphere–sphere intersections which need to be investigated. Our algorithm is much faster in the VDW surface area calculation because the number of sphere–sphere intersections is much smaller in this case. The same relativities should hold for smaller or less compact molecules.

Is it possible to calculate the molecular surface area faster? Our algorithm requires only a minimum number of points to be tested by calculation of the scalar product. Only points in the shadowed slice (Fig. 1) need to be checked for each intersection. This makes this algorithm the fastest of possible algorithms which use some kind of a selection of the points which should be tested explicitly. Approaches based on different strategies, such as that of ref. 4, may, at least in principle, be faster. It should be noted, however, that there is always a place for better and/or machine-specific implementations.

Finally, some comments should be made about memory requirements. The most memory-hungry algorithm is that of ref. 2. In our implementation it required $12.5N^2$ bytes for the table of masks. This is an improvement on the $25N^2$ bytes in the original implementation,[2] but leads to 50 MB memory for 2048 points, which prohibits its use on most current workstations. The implementation of our algorithm required $4N^2$ bytes for the sorted table of cosines (we used 2 bytes to store the point number, and 2 bytes to store the cosine). This translates to 16 MB of memory for 2048 points. Also, about 1 MB is required for the look-up table of points numbers for 2048 points. Memory requirements for the algorithm of ref. 4 are negligible.

Considering the discussion above and the data of Table I, we can make the following practical recommendations on the choice of algorithm. If fast but not very accurate estimation of the molecular surface area with a small number of points is all that is needed, then the algorithm of ref. 2 should be chosen. For more precise calculations of VDW surface, our algorithm can be recommended. Accurate evaluation of the solvent accessible surface area of compact molecules can be done approximately equally fast using our algorithm or the approach of ref. 4. Our algorithm ought to perform better for less compact molecules. It should also be noted that the order of loops in our algorithm allows it to be easily combined with current molecular mechanics/molecular dynamics codes and this may provide an additional advantage over the method in ref. 4.

Conclusion

A fast and accurate method for numerical molecular surface area evaluation has been described and the speeds of this algorithm and two from the literature are compared. The algorithm of ref. 2 has been shown to be the fastest, but, unfortunately, due to the inherent inexactitude of this algorithm, a two- or three-fold increase in the number of points is needed to achieve the same average deviation as for our algorithm. Our method and the approach of ref. 4 are similarly fast when calculating the SA surface area of large compact molecules, although our approach tends to be faster as the number of points increases. Also, our algorithm is clearly superior for calculating the VDW surface area.

Acknowledgments

The authors thank Dr. F. Eisenhaber for making his program NSC[4] available for the calculations. Financial support from the National Health and Medical Research Council (NH & MRC) is gratefully acknowledged.

References

1. A. A. Bliznyuk and J. E. Gready, *J. Comput. Chem.*, **17**, 962 (1996), preceding article.
2. S. M. Le Grand and K. M. Merz, Jr., *J. Comput. Chem.*, **14**, 349 (1993).
3. R. Abagyan, M. Totrov, and D. Kuznetsov, *J. Comput. Chem.*, **15**, 488 (1994).
4. F. Eisenhaber, P. Lijnzaad, P. Argos, C. Sander, and M. Scharf, *J. Comp. Chem.*, **16**, 273 (1995). Program NSC is available from http://www.embl-heidelberg.de/argos/ASC.21/eisenhaber-home.html.
5. J. Tomasi and M. Persico, *Chem. Rev.*, **94**, 2027 (1994).
6. A. A. Rashin, *Prog. Biophys. Mol. Biol.*, **60**, 73 (1993).
7. M. C. Surles, J. S. Richardson, D. C. Richardson, and F. P. Brooks, Jr., *Protein Sci.*, **3**, 198 (1994).
8. A. Shrake and J. A. Rupley, *J. Mol. Biol.*, **79**, 351 (1973).
9. F. M. Richards, *Ann. Rev. Biophys. Bioeng.*, **6**, 151 (1977).
10. T. Furuki, A. Umeda, M. Sakurai, Y. Inoue, R. Chujo, and K. Harata, *J. Comput. Chem.*, **15**, 90 (1994).
11. V. Yip and R. Elber, *J. Comput. Chem.*, **10**, 921 (1989).
12. F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer, Jr., M. D. Brice, J. D. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi, *J. Mol. Biol.*, **112**, 535 (1977).
13. A. Bondi, *J. Phys. Chem.*, **68**, 441 (1964).